



INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

INTEGRANTES:

SANLUIS CASTILLO JOSE DAVID

MATERIA:

WEB APPLICATIONS DEVELOPMENT

PROFESOR:

CIFUENTES ALVAREZ ALEJANDRO SIGFRIDO.

"PRACTICA: USO DE GET, POST, DOGET(), DOPOST()"

GRUPO: 3CM2

Uso de GET, POST, doGet(), doPost().

Introducción.

Un servlet es una clase que implanta la interfaz Servlet, que se encuentra dentro del paquete `javax.servlet` y hereda de la clase `GenericServlet`. `GenericServlet` es independiente del protocolo utilizado, si el servlet solamente utiliza HTTP se puede heredar de la clase `HttpServlet`. Los servlets generan páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web:

- Un usuario solicita una página HTML, JSP, o un servlet.
- El servidor carga e inicializa el servlet, llamando a `init()`.
- El servlet maneja cero o más peticiones del cliente, llamando a `doGet()`, `doPost()`, u otro método.
- El servidor elimina el servlet, llamando a `destroy()`.

Desarrollo.

Observaciones:

Para iniciar con el código del servlet, se indican los paquetes necesarios:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

La clase que implante el servlet debe heredar de `javax.servlet.HttpServlet`, o de `javax.servlet.http.HttpServlet`, de la siguiente manera:

```
public class ServletGetPost extends HttpServlet{
    :
    public void init(){
        System.out.println("Se inicializa el servlet");
    }
    public void init(ServletConfig conf) throws ServletException{
        super.init(conf);
    }
    :
}
```

En esta clase se debe codificar al menos un método, por ejemplo `doGet()`:

```
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException{
    :
}
```

Los dos parámetros representan los flujos de datos intrínsecos de petición y respuesta.

Se define una clase `PrintWriter` para asociar el flujo de salida:

```
PrintWriter out = response.getWriter();
String a, b;
int x, y, resultado;
a = request.getParameter("a");
b = request.getParameter("b");
```

, donde `getParameter()` regresa una cadena. Para manejar los resultados:

```
try{
    x = Integer.parseInt(a);
    y = Integer.parseInt(b);
    resultado = x + y;
} catch(Exception e){
    e.printStackTrace();
    out.println("Error al recibir parámetros con GET");
    out.print(e);
    out.close();
    return;
}
```

Se indica el tipo de contenido MIME con "text/html", "images.jpeg" o "application/x-gzip":
`response.setContentType("text/html");`

Para la apariencia de la página web se utiliza el método `println()`:

```
out.println("<html>");
out.println("<head><title>Un servlet básico</title></head>");
out.println("<body>");
out.println("<h1>Esta es una prueba de un servlet con GET</h1>");
out.println("<h1>La suma de " + x + " + " + y + " = " + resultado + "</h1>");
out.println("</body></html>");
out.close();
}
```

El código de `doPost()` es el siguiente:

```
public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException,
IOException{
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String a, b;
    int x, y, resultado;
    a = request.getParameter("a");
    b = request.getParameter("b");
    try{
        x = Integer.parseInt(a);
        y = Integer.parseInt(b);
        resultado = x - y;
    } catch(Exception e){
        e.printStackTrace();
        out.println("Error al recibir parámetros con POST");
        out.print(e);
        out.close();
        return;
    }
    out.println("<html>");
    out.println("<head><title>Un servlet básico</title></head>");
    out.println("<body>");
    out.println("<h1>Esta es una prueba de un servlet con POST</h1>");
    out.println("<h1>La resta de " + x + " - " + y + " = " + resultado + "</h1>");
    out.println("</body></html>");
    out.close();
}
```

Los métodos se pueden utilizar desde un archivo HTML.

Para el método GET, el código de `doGet.html` es:

```
<html>
<head><title>Servlet con GET</title></head>
<body>
<h1>Suma de dos enteros</h1>
<form action="/ServletGetPost" method="GET">
<p>Numero a = <input type="text" name="a"></p>
<p>Numero b = <input type="text" name="b"></p>
<p><input type="submit" value="Hacer suma"></p>
</form>
</body></html>
```

Para el método POST, el código de `doPost.html` es:

```
<html>
<head><title>Servlet con POST</title></head>
<body>
<h1>Suma de dos enteros</h1>
<form action="/ServletGetPost" method="POST">
<p>Numero a = <input type="text" name="a"></p>
<p>Numero b = <input type="text" name="b"></p>
<p><input type="submit" value="Hacer resta"></p>
</form>
</body></html>
```

Es importante que el valor del atributo `name` sea el mismo utilizado en el método `getParameter()` en el servlet.

También en el código del servlet se sobrecargan los siguientes métodos:

```
public String getServletInfo(){
    return "Descripcion sencilla del servlet";
}

public void destroy(){
    System.out.println("Servlet destruido");
}
```

Se configura un descriptor de despliegue en un archivo **web.xml**, que indica los servlets de la aplicación web y la URL en la cual se atienden las peticiones.

```
<?xml versión="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<!--Información -->
<web-app>
    <description>Configuración de un servidor de aplicaciones</description>
    <display-name> Ejemplo de un Servlet</display-name>
    <servlet>
        <servlet-name>ServletGetPost</servlet-name>
        <servlet-class>ServletGetPost</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ServletGetPost</servlet-name>
        <url-pattern>ServletGetPost</url-pattern>
    </servlet-mapping>
</web-app>
```

La etiqueta `<servlet-mapping>` indica que para llamar al servlet se hace a través de un directorio virtual `/servlet/`, por ejemplo, `<url-pattern>/servlet/ServletGetPost</url-pattern>`.

Tomcat buscará el archivo `ServletGetPost.class` dentro de la carpeta `classes`.

Ejecución del servlet:

El servlet se puede acceder en las siguientes formas:

`http://servidor:8080/ServletGetPost` para el archivo JSP

Si el servlet se guarda en otra carpeta:

`http://localhost:8080/servlets/servlet/ServletGetPost`

Si el servlet se invoca desde una página HTML:

`http://servidor:8080/doGet.html` para el archivo HTML

`http://servidor:8080/doPost.html` para el archivo HTML

Algunos métodos de la clase `HttpServlet` son:

`init()`, llamado solamente en la inicialización del servlet.

`destroy()`, llamado cuando se destruye la instancia del servlet.

`service()`, no se puede sobrecargar.

`doGet()`, `doPost()`, `doPut()`, `doDelete()`, `doOptions()`, `doTrace()`, llamados según el tipo de petición HTTP.

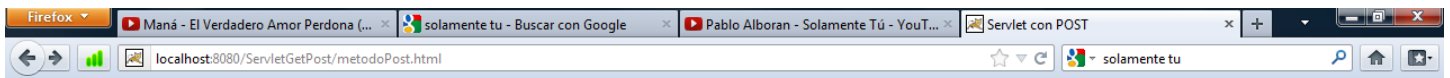
`getServletInfo()`, regresa información sobre el servlet.

`getServletName()`, regresa el nombre del servlet, independientemente del nombre de la clase.

`getInitParameter()`, regresa los parámetros de inicialización.

`getInitParameterNames()`, regresa una enumeración de los nombres de esos parámetros.

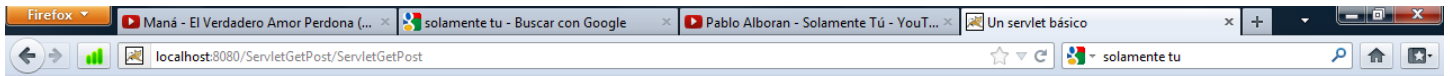
EJERCICIO CON EL METODO POST:



Suma de dos enteros

Numero a =

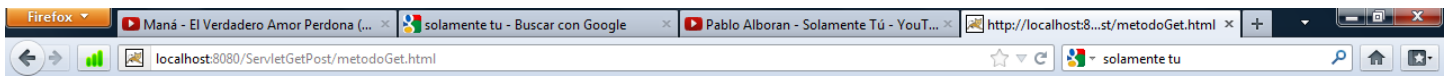
Numero b =



Esta es una prueba de un servlet con GET

La suma de $5 + 5 = 10$

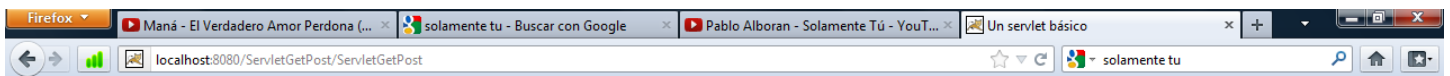
EJERCICIO CON EL METODO GET:



Suma de dos enteros

Numero a =

Numero b =



Esta es una prueba de un servlet con GET

La suma de $7 + 17 = 24$
